Genetic Design of Real-Time Fuzzy Logic Controllers

Kuan-Shiu Chiu* and Andrew Hunter

School of Computing and Information Systems

University of Sunderland

England, UK

cs0ksc@isis.sunderland.ac.uk and cs0ahu@isis.sunderland.ac.uk

*** also Computer Center, Tamsui Oxford University College, Taiwan**

Abstract

This paper discusses the use of Genetic Algorithms (GAs) to design Fuzzy Logic Controllers for Real-Time control of flows in sewerage networks. Genetic Algorithms act as an expert to define fuzzy rule bases and membership functions. The GA usually uses binary chromosome for coding solutions. This paper proposes using floating-point chromosomes to design the fuzzy logic controller using SUGAL, The SUnderland Genetic Algorithms Library. This coding provides schemata of shortest defining length, one for each parameter, which prevents disruption by crossover. Our choice of design is briefly described. GAs are reinforcement learning techniques based on a trial and error, requiring heavy computation. A simulator is needed for the system. We use the C language to write the simulator, running on Sun SPARCstation 5 machines. The comparison between this approach and various fixed penstock control settings, and genetically-designed Neural Networks, is discussed. This paper reports experiments demonstrating that GAs are both effective and robust to design fuzzy logic controllers in sewerage networks control problems.

Keywords

Genetic Algorithms, Fuzzy Logic Controller, Real-Time Control, Reinforcement Learning, Flow Systems.

1. Introduction

Combined sewerage systems are used in many cities and countries. The same pipes carry foul and storm flows in the systems. Most of the time they function normally. During heavy rainfall the system faces the problem of overflows which occur when the system cannot carry all the flows entering the sewers. Construction of new storage and sewers could be the solution to completely eliminate overflow problems. However, such schemes are expensive and can also be extremely disruptive because the sewerage networks may extend across wide geographical areas. Since inflows are seldom constant across an entire system, we can reduce the overflow spills by controlling the flow between parts of the system which are under different loading. This may be achieved by the installation of automatically actuated penstocks which can be opened or closed to control the flow past a certain point [1].

Standard optimization techniques, such as linear and dynamic programming, have been applied to these type of problems, but without great success other than for very simple networks[2]. Linear programming is applied in simplified problems, but alternative solutions are difficult to evaluate. Dynamic programming could be successful if all possible configurations are tested, which requires sufficient computing resources. For a complex systems, it is unrealistic.

In this paper, we implement fuzzy logic controllers for the problem. Genetic Algorithms are used as an expert to design the system. Our simulator is tested on two tank and three tank systems.

2. Fuzzy Logic and Genetic Algorithms

To implement the controllers, there are a wide range of choices. These are Neural Networks, Fuzzy Logic Systems, Genetic Programming, and Classifier Systems [1]. The Choice of paradigm must satisfy the two factors:

1. Control penstock setting.
2. Minimise the amount of spillage across the systems.

The controllers must be able to adjust the penstock setting according to the inflow, current tank level, and the next adjacent tank level. Real-valued processing is required to represent the three inputs and one output, making neural networks and fuzzy logic systems a sensible choice. Both of them are well-suited to handle real values.

To optimise the controllers, a learning algorithm is required. Reinforcement learning methods, such as Genetic Algorithms, are feasible solutions. Genetic design of neural network controllers and Genetic design of fuzzy logic controllers are the two paradigms which satisfy the two required factors of the controllers.

A previous paper [1] shows how GAs can be used to optimise the weights of Neural Network controllers. In this paper, we focus on Genetic Design of Real-Time Fuzzy Logic Controllers.

3. Genetic Design of Fuzzy Logic Controllers

The GA is used to design fuzzy logic controllers (GA-FLC) in the following ways:

1. Tuning the membership functions of a fuzzy set.
2. Defining the rule base.
3. Tuning the membership functions and defining the rule base at the same time.

There is a great deal of research in this area [3]. This paper deals with the third approach. Most researchers have used binary chromosomes for coding solutions and map binary into integer or floating point. However, this is not a natural choice of representation. The floating-point representation has been proved to be faster, more consistent, and providing a higher precision[4]. It also has the following advantages [5] : (1) comfort with one-gene-one-variable correspondence, (2)

avoidance of Hamming cliffs and other artifacts of mutation operating on bit strings treated as unsigned binary integers, (3) fewer generations to population conformity, (4) reduction of the opportunities for normal-mode deception.

We use real-coded chromosomes for coding solutions by using SUGAL, The SUnderland Genetic Algorithms Library [6]. Each floating-point chromosome contain membership functions and the rule base. They are described in the next two sections.

3.1 The membership functions

There are three inputs and one output for the controllers: inflow, current tank level, next adjacent tank level for inputs, and the penstock setting for the output. Four fuzzy linguistic variables are required for the inputs and the output. For simplicity and efficiency, each linguistic variable has three fuzzy sets: small, medium, and large. Triangular fuzzy sets are constructed. The two slopes of each fuzzy set are symmetric. The first slope of the first fuzzy set and the second slope of the third fuzzy set are 0. Two parameters are used for each fuzzy variable (see Figure 1). The first parameter, x, is for the anchor point of the first and third fuzzy sets. The second parameter, y, is for the second fuzzy set.

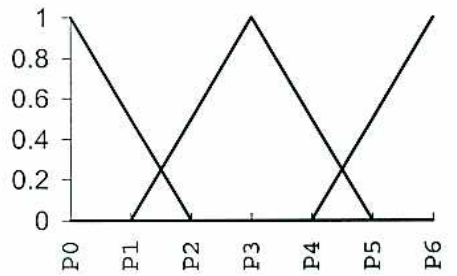

Figure 1 Defining Membership functions

3.2 The rule base

There are two approaches to the definition of the fuzzy rule base using GAs: position-dependent coding and context-dependent coding. In position-dependent coding, each possible fuzzy rule is represented at a fixed locus on the chromosome. This suffers from the "curse of dimensionality" for complex controllers. The context-dependent coding, which uses messy GAs [7], avoids the "curse of dimensionality" by representing a subset of possible rules in a locus-independent fashion. However, there are problems with under-specification and over-specification of the rule base, and convergence may be slow. Since our problem has only a small number of variables, we use the simple position-dependent scheme. Each gene in the chromosome represents a particular combination of fuzzy input sets. The gene value indicates the fuzzy output sets. Each gene is a floating-point value between 0 and 1. The range of a gene value is partitioned into 4 parts and mapped to 3 output fuzzy sets and "not used", respectively. The "not used" rules will not be evaluated. There are no more than 27 rules created (see Figure 2).

Chromosome	C	C	C	C	.....	C
Rule #	1	2	3	4	.....	27
Inflow	SM	SM	SM	SM	.....	
Current Tank Level	SM	SM	SM	MD	.....	LG
Next Tank Level	SM	MD	LG	SM	.....	LG

C = {SM, MD, LG, Not used}

Figure 2 Rule base part of Chromosome

3.3 Fitness function

When applying GAs, two things are problem dependent: the encoding of the problem into a chromosome and the fitness function. The former is described in the previous two sections. The fitness of each chromosome is equal to the number of overflows. We run 100 different weather

simulations. Each chromosome is tested against the 100 simulations. The number of overflows is used as the fitness. The system architecture for GA-FLC is showed in Figure 3.

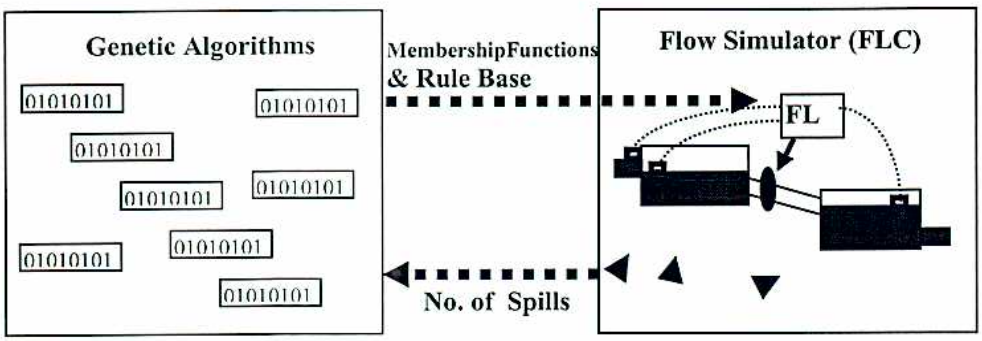

Figure 3 The GA-FLC System Architeture

4. Experiments

Our experiments are tested on several Sun SPARCstation 5 machines. We use the C language to write the simulator and fuzzy logic controller integrated with SUGAL. The Min-Max-COG is used for fuzzy reasoning. Before the experiments, we briefly discuss the simulations.

4.1 Simulation design

A typical Genetic Algorithm for a reasonably simple problem might involve a population of one hundred chromosomes run for one hundred generations, requiring a total of 10,000 chromosome evaluations. One hundred inflow sequences were designed to simulate the different weather conditions. Each simulation has about 500 time-steps. On each time-step a fuzzy controller in the system is executed and a set of flow-equation is solved. So we have about 50,000 time-steps for each of 10,000 chromosome evaluations, resulting in approximately 500,000,000 simulation time-steps in total.

For the smallest sewage network systems, 2 tanks, it requires millions of time-steps simulations to train. The computation time is extremely high. However, once the controllers are trained, they can be executed in the required time for real-time control (i.e. tens of seconds) on any microprocessor.

4.2 Two tank experiments

The first group of experiments is done by using the two tank system (see Figure 3). This system has two interconnected tanks, with a inflow to the upper tank and an outflow from the lower. A penstock is located on the jointing pipe. A spill is deemed to occur if the water level in either tank passes the top. 100 simulations are tested using the following benchmarks.

1. The optimal spillage rate.
2. Various fixed penstock setting.
3. Genetic design of Fuzzy Logic Controllers.
4. Genetic design of Neural Network controllers (discussed in reference [1]).

Since water can only exit the system through the outflow in the lower tank and the speed of outflow is related to the depth in the lower tank, to get the optimal spillage rate, we maximise the rate of outflow by keeping the lower tank as full as possible. This is done by initially opening the penstock fully, then as the level of the bottom tank nears the top, opening the penstock just sufficiently to keep it fully topped up.

Our second experiment for the two tanks system fixes the penstock to a particular setting throughout the 100 simulations. The fixed penstock settings are tested from 0 to 100% open in steps on 1%. Not surprisingly, the largest number of spills (78%) occurs when the penstock is full closed

throughout the simulations. However, the fully open penstock produces 49% spillage, which is not the lowest number. The lowest number of spills (38%) results from a fixed aperture of 39%. The results are showed in Figure 4.

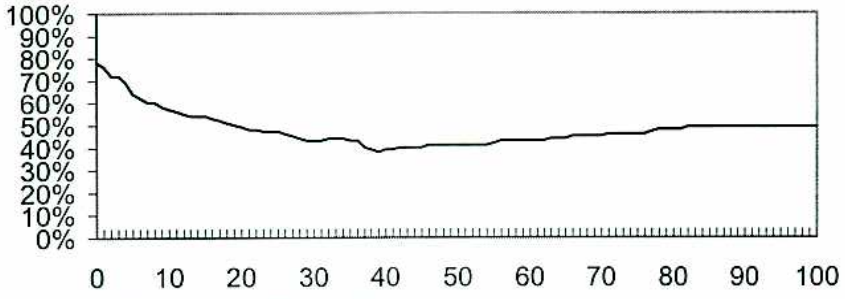

Figure 4 Two Tanks Fixed Aperture Flood Profile

For the GA-FLC, we use 3 sensor inputs; the inflow to the upper tank and the levels in the two tanks. The only output is the aperture setting (see Figure 3). The results show GA-FLC has 29% spills which is almost the optimal value. The performance of GA-NN controller is similar to GA-FLC. Figure 5 displays the comparisons of our benchmarks.

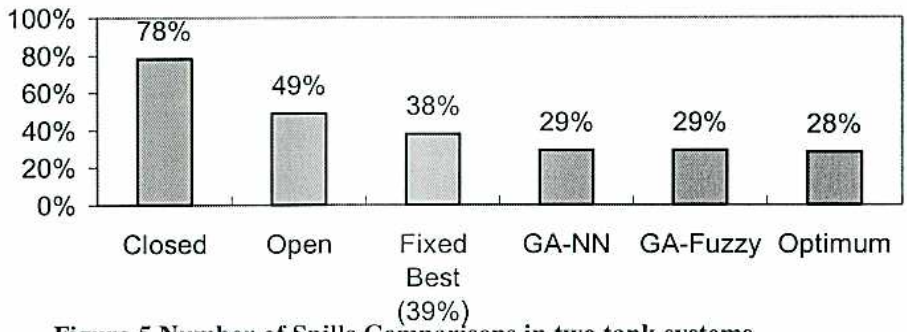

Figure 5 Number of Spills Comparisons in two tank systems

4.3 Three tank experiments

The three tank system is used for second group of experiments (see Figure 6). The system contains three interconnected tanks, two up-stream tanks feeding into a single downstream tank. The optimal strategy in the two tank system can no longer be used, since one up-stream tank could have a heavy inflow and the other a light inflow. In this case the second tank should close its penstock to allow the first tank to empty more rapidly through the shared lower tank. However, if both upper tanks are under lower pressure, they should be closed to clear the lower tank.

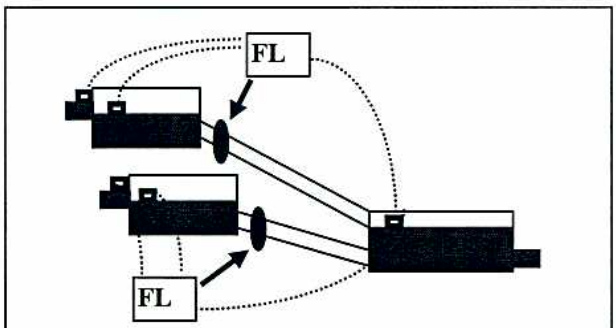

Figure 6 Three Tanks System

We use the same 100 inflow simulations for the two up-stream tanks. In order to simulate a weather pattern moving across a geographical area and increasing in intensity, a delay and scaling factor is used for the inflow into the second up-stream tank.

The results are showed in Figure 7. We note that the GA-FLC outperforms the best fixed aperture setting.

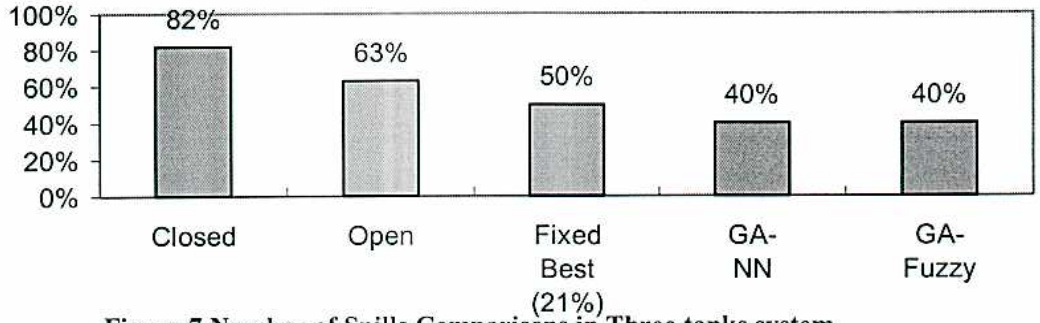

Figure 7 Number of Spills Comparisons in Three tanks system

4.4 The comparisons with GA-NN

The GA design of neural network controllers for the same problem has previously been discussed [1]. The performance is similar to GA-FLC. However, GA-FLC converges more quickly, probably because the search space is more constrained. However, when the number of inputs increases, our coding approach will cause an exponential increase in the number of rules generated , in which case the advantage over the GA-NN approach may be lost.

5. Conclusion

The power of fuzzy logic systems is based on its tolerance of imprecision and ambiguous information. It is well-suited for a complex problem, like real-time control of flow systems. An expert is required to design the system. In this paper, we use the reinforcement learning abilities of the Genetic Algorithms to do the design. The results demonstrate that this is a very good combination.

As discussed in section 4.1, the smallest sewage system requires millions of time-step simulations to train. For our experiments, it needs many hours to run on Sun SPARCstation 5 machines. A more powerful machine could be a better choice.

6. References

[1] Hunter, Andrew (1997) "Genetic Design of Real-Time Neural Network Controllers." Neural Computing & Applications. V4. N3. 1997. Springer-Verlag.

[2] Petersen. SO. (1987) Real Time Control of Urban Drainage Systems. M.sc. Thesis. Dept. of Environmental Engineering, Technical University of Denmark, August 1987.

[3] Cordón O., Herrera E., and Lozano M. (1996) "A Classified Review on the Combination Fuzzy Logic-Genetic Algorithms Bibliography." Technical Report DECSAI-95129, Dept. of Computer Science and A.I., University of Granada, December 1996.

[4] Michalewicz, Zbigniew (1996) Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, USA.

[5] Goldberg, David E. (1990). "Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking" Department of General Engineering, University of Illinois at Urbana-Champaign. Technical Report No. 90001, September 1990.

[6] Hunter, Andrew The SUGAL Genetic Algorithm Simulator. http://osiris.sund.ac.uk/ahu/sugal/home.html.

[7] Hoffman F. and Pfister G. (July 1996) "Evolutionary learning of a fuzzy control rule base for an autonomous vehicle." In Proc. Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU'96), pages 1235--1240. Granada.